Zastosowania procesorów sygnałowych – projekt 2020

1. Generowanie sygnałów na DSP

Autor: Grzegorz Szwoch, greg@multimed.org, Katedra Systemów Multimedialnych, WETI PG

Aktualizacja: 10.04.2020

Wstęp

W tym zadaniu projektowym studenci będą tworzyli funkcje generujące podstawowe sygnały testowe z wykorzystaniem symulatora DSP. Sygnały testowe będą użyteczne do badania algorytmów opracowanych w ramach następnych zadań projektowych.

Zadaniem studentów jest:

- wykonanie poleceń opisanych w kolejnych punktach instrukcji,
- opracowanie uzyskanych wyników w formie raportu (każdy punkt instrukcji wymienia co powinno znaleźć się w raporcie, ponadto zaleca się zamieszczenie w raporcie własnych wniosków),
- umieszczenie w serwisie sprawozdań skompresowanego pliku, zawierającego: raport w formie pliku PDF oraz opracowany kod (plik *main.c*, ew. inne pliki dodane lub zmodyfikowane przez studenta); kod powinien być oczyszczony z niepotrzebnych rzeczy, czytelnie sformatowany i opatrzony komentarzami.

Materiały pomocne w wykonywaniu zadania to prezentacje z wykładów ZPS: *Generowanie sygnałów* oraz *Systemy liczbowe*, a także dokumentacja biblioteki DSPLIB (dostępna w materiałach pomocniczych na stronie Katedry oraz po wpisaniu SPRU422J w przeglądarce internetowej).

Przed rozpoczęciem wykonywania zadań należy wykonać w CCS kopię szablonu projektu i nazwać ją *Generatory*. Wszystkie polecenia należy wykonać w tym projekcie.

1. Generowanie sygnału piłokształtnego

Zadania:

- Napisać funkcję *saw*, generującą sygnał piłokształtny. Funkcja powinna przyjmować trzy parametry: (1) wskaźnik do tablicy typu *int*, (2) liczbę elementów w tablicy, (3) wartość kroku amplitudy, którą należy obliczyć na podstawie częstotliwości sygnału. Funkcja powinna wypełnić tablicę wartościami próbek sygnału o żądanej częstotliwości.
- Przyjąć częstotliwość sygnału równą 100 Hz i częstotliwość próbkowania 48000 Hz. Obliczyć wartość kroku w formacie Q15, zapisać ją w kodzie jako globalną stałą.
- W funkcji *main*, wywołać napisaną funkcję, generując 5000 próbek sygnału. Przedtem utworzyć globalną tablicę na próbki. Rozmiar tablicy powinien być zapisany jako globalna stała i tej stałej należy używać przy wywoływaniu funkcji.

- Zbudować i uruchomić program. Zatrzymać program na linii *while (1)*.
- Wykonać wykres czasowy tablicy z próbkami oraz wykres widma sygnału w tablicy (dla rozmiaru FFT równego 2048).

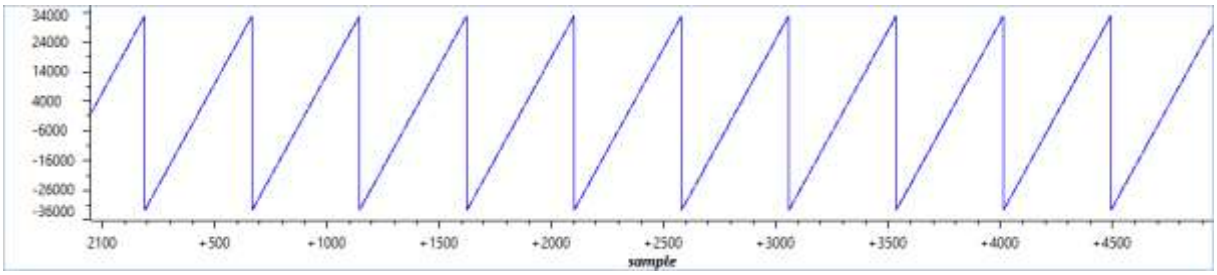
W raporcie:

- Podać obliczenia wartości kroku amplitudy.
- Zamieścić kod funkcji *saw*. Tekst kodu powinien być wstawiony z użyciem czcionki o stałej szerokości, np. Consolas. Dopuszczalne jest wstawienie zrzutu ekranu, ale proszę przyciąć obraz do istotnego fragmentu.
- Zamieścić wykres czasowy. Ocenić poprawność uzyskanych wyników.
- Zamieścić wykres widma (można powiększyć skalę częstotliwości do istotnego zakresu). Opisać kształt widma.
- Obliczyć i podać rzeczywistą wartość częstotliwości uzyskanego sygnału. Dlaczego nie możemy uzyskać częstotliwości dokładnie 100 Hz?
- Zaproponować metodę generowania „odwróconej piły” (opadające zbocze).

Podpowiedzi

Sygnał piłokształtny jest bardzo prosty do wytworzenia. Stosujemy akumulator, czyli zmienną, do której co próbkę dodajemy stałą wartość kroku amplitudy, zależną od częstotliwości sygnału. Tworzymy wewnątrz funkcji zmienną akumulatora (typ *int*) i inicjalizujemy ją na zero. Dobrze jest zadeklarować tę zmienną z przedrostkiem *static*, dzięki temu będzie ona pamiętała swoją wartość przy kolejnych wywołaniach funkcji. Następnie tworzymy pętlę po indeksach tablicy, umieszczamy w bieżącej komórce tablicy aktualną wartość akumulatora, po czym zwiększamy akumulator o wartość kroku. Efekt przepełnienia zakresu jest w tym przypadku pożądany.

Sposób obliczania wartości kroku był podany na wykładzie. Można go obliczyć następująco. W ciągu jednego okresu sygnału, amplituda ma zmienić się o pełny zakres, czyli o 2^{16}. W ciągu jednej sekundy ma być 100 takich okresów. Częstotliwość próbkowania wynosi 48000 próbek na sekundę. To wszystko co jest potrzebne.



2. Generowanie sygnału prostokątnego

Zadania:

- Napisać funkcję *rect*, generującą sygnał prostokątny. Funkcja powinna przyjmować cztery parametry: (1) wskaźnik do tablicy typu *int*, (2) liczbę elementów w tablicy, (3) wartość kroku

amplitudy, (4) wartość progową, która określa proporcje dodatniej części okresu do ujemnej. Funkcja powinna wypełnić tablicę wartościami próbek sygnału o częstotliwości jak w poprzednim przypadku.

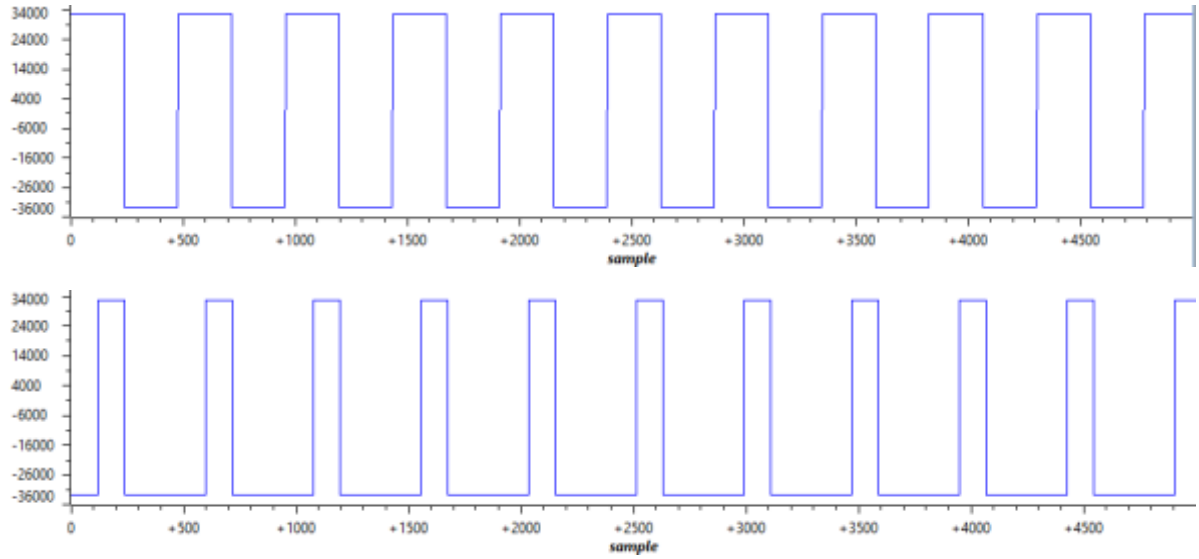
- W funkcji *main*, zakomentować poprzednią funkcję (stawiając znaki // na początku linii), następnie wywołać napisaną funkcję, generując 5000 próbek sygnału dla wartości progowej równej zero.
- Zbudować i uruchomić program. Zatrzymać program na linii *while (1)*.
- Wykonać wykres czasowy tablicy z próbkami oraz wykres widma sygnału w tablicy (dla rozmiaru FFT równego 2048).
- Powtórzyć eksperyment, tym razem podając taką wartość progową, aby proporcje dodatniej części okresu do ujemnej były równe 1:3 (patrz przykład dalej w tym punkcie).

W raporcie:

- Zamieścić kod funkcji.
- Zamieścić wykresy czasowy. Ocenić poprawność uzyskanych wyników.
- Zamieścić wykresy widma. Opisać kształt widma, różnice w porównaniu z sygnałem piłokształtnym, różnice pomiędzy różnymi wartościami progowymi.

Podpowiedzi

Sygnał prostokątny można łatwo uzyskać metodą progowania sygnału piłokształtnego. Jeżeli aktualna wartość akumulatora jest dodatnia, generujemy wartość maksymalną, w przeciwnym razie – minimalną (w naszym przypadku odpowiednio 32767 i -32768). Przy wartości progowej równej zero uzyskujemy równe proporcje części dodatniej i ujemnej. Zmieniając wartość progową modyfikujemy te proporcje.



3. Generowanie sygnału trójkątnego

Zadania:

- Napisać funkcję *tri*, generującą sygnał trójkątny. Funkcja powinna przyjmować trzy parametry: (1) wskaźnik do tablicy typu *int*, (2) liczbę elementów w tablicy, (3) wartość kroku amplitudy. Częstotliwość 100 Hz, jak poprzednio.
- W funkcji *main*, zakomentować poprzednią funkcję i wywołać napisaną funkcję, generując 5000 próbek sygnału.
- Zbudować i uruchomić program. Zatrzymać program na linii *while (1)*.
- Wykonać wykres czasowy tablicy z próbkami oraz wykres widma sygnału w tablicy (dla rozmiaru FFT równego 2048).

W raporcie:

- Zamieścić kod funkcji.
- Zamieścić wykres czasowy. Ocenić poprawność uzyskanych wyników.
- Zamieścić wykres widma. Opisać kształt widma oraz różnice w porównaniu do poprzednio analizowanych sygnałów.

Podpowiedzi

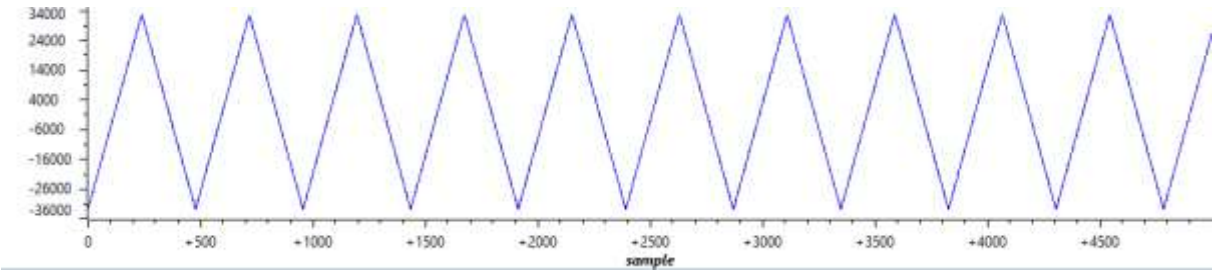
Sygnał trójkątny można obliczyć z sygnału piłokształtnego w następujący sposób:

$$y = 2 \cdot |x| - 1 = (|x| - 0{,}5) \cdot 2$$

Wartość bezwzględną w języku C można uzyskać następująco:

```
y = x < 0 ? -x : x;
```

Optymalny sposób mnożenia zmiennej przez 2 to wykonanie przesunięcia bitowego: (x << 1).



4. Generowanie sygnału sinus za pomocą szeregu Taylora

Zadania:

- Napisać funkcję *sint*, generującą sygnał sinusoidalny metodą szeregu Tylora (opis w dalszej części podrozdziału). Funkcja powinna przyjmować trzy parametry: (1) wskaźnik do tablicy typu *int*, (2) liczbę elementów w tablicy, (3) wartość kroku amplitudy. Częstotliwość 100 Hz, jak poprzednio.
- Uwzględnić składniki szeregu do siódmej potęgi włącznie.
- Stałe współczynniki szeregu należy zapisać wewnątrz funkcji jako stałe typu *long*.
- Obliczenia należy wykonać w sposób minimalizujący liczbę operacji mnożenia.

- Przed obliczeniem ostatecznego wyniku należy zastosować zaokrąglanie wartości (nie obcinanie).
- W funkcji *main*, zakomentować poprzednią funkcję i wywołać napisaną funkcję, generując 5000 próbek sygnału.
- Zbudować i uruchomić program. Po uruchomieniu, wybrać z menu CCS: *Run* > *Clock* > *Enable*. Zatrzymać program na linii *while (1)*. Zanotować liczbę cykli zużytą przez funkcję – jest ona wyświetlana na dolnym pasku stanu, obok ikony zegara.
- Wykonać wykres czasowy tablicy z próbkami oraz wykres widma sygnału w tablicy (dla rozmiaru FFT równego 2048).

W raporcie:

- Zamieścić kod funkcji.
- Zamieścić wykres czasowy. Ocenić poprawność uzyskanych wyników.
- Zamieścić wykres widma. Opisać kształt widma. Na podstawie widma ocenić jak „czysty" (pozbawiony zniekształceń) jest generowany sygnał.
- Wyjaśnić dlaczego w tej implementacji nie jest zasadne uwzględnianie dalszych składników szeregu Taylora, tzn. dziewiątej potęgi i dalszych.

Podpowiedzi

To zadanie jest trudniejsze od poprzednich, dlatego omówimy je bardziej szczegółowo. Rozwinięcie funkcji sinus w szereg Taylora ma następującą formę:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

gdzie *x* jest wartością kąta fazowego, wyrażoną w radianach. W zakresie kątów od -π/2 do π/2 wystarczą cztery składniki szeregu, jak w powyższym wzorze, jednak w pozostałym zakresie kątów dokładność jest niewystarczająca. Można jednak wykorzystać fakt, że okres sygnału sinus składa się z czterech ćwiartek, które są nawzajem symetryczne. Wystarczy więc obliczyć wartości dla kątów z przedziału od 0 do π/2, a stąd łatwo już uzyskać pozostałe trzy ćwiartki.

Na początek należy obliczyć stałe wartości współczynników szeregu i zapisać je jako stałe w kodzie funkcji, najlepiej jako typ *long* (co ułatwi późniejsze operacje). W powyższym wzorze, kąt fazowy *x* zmienia się w zakresie od -π do π. Potrafimy wygenerować sygnał fazy, zrobiliśmy to już w jednym z poprzednich zadań. Sygnał ten przyjmuje jednak wartości od -1 do 1 (dziesiętnie), więc współczynniki należy przeskalować, mnożąc je przez odpowiednie potęgi π:

$$a_1 = \pi$$

$$a_3 = -\pi^3 / 3!$$

$$a_5 = \pi^5 / 5!$$

$$a_7 = -\pi^7 / 7!$$

$$y = a_1x + a_3x^3 + a_5x^5 + a_7x^7$$

Ponieważ zakres Q15 jest za mały aby zapisać wartości współczynników, musimy użyć formatu Q12 (inaczej Q4.12). Wartości kąta fazowego pozostają w zapisie Q15.

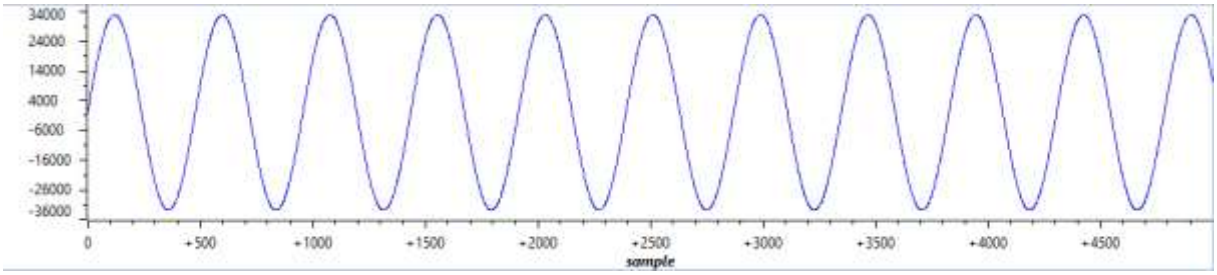
Teraz pętla obliczająca wartości próbek. Najpierw sprowadzamy wartość kąta fazowego do pierwszej ćwiartki. Obliczamy wartość bezwzględną kąta fazowego. Jeżeli wartość ta jest większa niż 0,5, zamieniamy ją na (1 – *x*), czyli (32767 – x) w Q15.

Następnie musimy obliczyć potęgi argumentu. Aby zminimalizować liczbę operacji mnożenia, można zauważyć, że: $x^3 = x^2 \cdot x$, $x^5 = x^2 \cdot x^3$, $x^7 = x^2 \cdot x^5$. Mnożenie najwygodniej jest wykonać za pomocą instrukcji *_smpy*. Wyniki należy zapisać w zmiennych typu *int*.

Aby obliczyć wyrażenie szeregu Taylora, trzeba już posłużyć się typem *long* (normalne operacje mnożenia i dodawania). Wynik obliczeń będzie zapisany w formacie Q27 (15 + 12). Aby przejść do zapisu Q15, trzeba wykonać przesunięcie bitowe w prawo. Ale przed tym musimy wykonać zaokrąglenie. Robi się to dodając jedynkę na najstarszej z pozycji z tych, które zostaną usunięte w wyniku przesunięcia bitowego. Przykład był prezentowany na wykładzie. Obliczony wynik rzutujemy do typu *int*.

Pozostaje do wykonania jeszcze jedna rzecz. Jeżeli oryginalna wartość kąta fazowego była ujemna (trzecia lub czwarta ćwiartka), musimy odwrócić znak wyniku. Na koniec zapisujemy wynik do tablicy.

Metoda może wydawać się skomplikowana, ale autor instrukcji wykonał implementację opisywanej funkcji w 24 liniach kodu, więc sam algorytm nie jest bardzo złożony.



5. Generowanie sygnału sinus za pomocą funkcji z DSPLIB

Zadania:

- W dokumentacji biblioteki DSPLIB znaleźć funkcję generującą próbki sygnału sinusoidalnego.
- W funkcji *main*, zakomentować poprzednią funkcję i wywołać funkcję z DSPLIB, generując 5000 próbek sygnału (częstotliwość jak poprzednio).
- Zbudować i uruchomić program. Tak jak poprzednio, zmierzyć i zanotować liczbę zużytych cykli procesora po zatrzymaniu programu na linii *while (1)*.
- Wykonać wykres czasowy tablicy z próbkami oraz wykres widma sygnału w tablicy (dla rozmiaru FFT równego 2048).

W raporcie:

- Zamieścić wykres czasowy i widmowy.
- Porównać wyniki z otrzymanymi w wyniku działania napisanej przez siebie funkcji.

- Podać i porównać zmierzone liczby cykli procesora dla obu implementacji. Wyjaśnić dlaczego jedna z implementacji działa dużo szybciej od drugiej, mimo że obie wykorzystują metodę szeregu Taylora do generowania sygnału.

Podpowiedzi

W praktyce nie będziemy pisali własnych implementacji generatora sygnału sinus. Biblioteka DSPLIB zawiera najczęściej używane procedury cyfrowego przetwarzania sygnałów. Dokumentacja (w języku angielskim) znajduje się na stronie Katedry w materiałach pomocniczych do projektu. Można ją również znaleźć wpisując SPRU422J w wyszukiwarce internetowej.

Biblioteka DSLIB używa własnego typu *DATA*, który na naszym procesorze jest synonimem typu *int*. Dla kompilatora wskaźnik *int** nie jest jednak tym samym co wskaźnik *DATA**. Dlatego musimy stosować rzutowanie wskaźników na obiekty typu *int*: (DATA*)x.

6. Generowanie białego szumu

Zadania:

- W funkcji *main*, zakomentować poprzednią funkcję i wywołać funkcję z DSPLIB, generującą 5000 próbek białego szumu. Należy pamiętać o zainicjalizowaniu algorytmu
- Zbudować i uruchomić program. Zatrzymać programu na linii *while (1)*.
- Wykonać wykres czasowy tablicy z próbkami oraz wykres widma sygnału w tablicy (dla rozmiaru FFT równego 2048).

W raporcie:

- Zamieścić wykres czasowy i widmowy. Opisać kształt sygnału i widma.

Podpowiedzi

Biały szum to sygnał o równomiernej lub gaussowskiej funkcji rozkładu prawdopodobieństwa oraz o płaskiej charakterystyce widma amplitudowego. Biały szum można wytworzyć za pomocą algorytmu generującego liczby pseudolosowe (RNG – *random numer generator*). Funkcja generatora jest zawarta w bibliotece DSPLIB. Przed pierwszym jej użyciem należy jednak dokonać inicjalizacji generatora – służy do tego specjalna funkcja. Jeżeli jej nie wywołamy, przy każdym uruchomieniu programu otrzymamy ten sam ciąg liczb.

Koniec zadania